Title	行列のスペクトル分解・固有ベクトルの分散計算 (Computer Algebra : Design of Algorithms, Implementations and Applications)
Author(s)	小原, 功任; 田島, 慎一
Citation	数理解析研究所講究録 (2009), 1666: 65-68
Issue Date	2009-10
URL	http://hdl.handle.net/2433/141077
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
第1666巻 2009年 65-68

行列のスペクトル分解・固有ベクトルの分散計算

小原功任
KATSUYOSHI OHARA
金沢大学*

田島慎一
SHINICHI TAJIMA
新潟大学†

1 行列のスペクトル分解

我々は, 行列のスペクトル分解に関して, 留数計算に基づいた新しいアルゴリズムを提案した. アルゴリズムの詳細については, 本講究録に掲載予定の田島・飯塚・樋口の論文に譲り, ここでは概略のみを述べるが, このアルゴリズムは並列化にも適しているものである. 本稿では, このアルゴリズムの並列化効率について報告する.

以下, A を有理数を要素とする n 次正方行列, $f(x)$ を A の最小多項式とし, $f(x)$ は無平方であると仮定する.

A の固有値 λ に対し,

$$P_\lambda = \frac{1}{2\pi\sqrt{-1}} \oint_\lambda (A - xE)^{-1} dx$$

は A のスペクトル分解を与える. すなわち

$$A = \sum_{f(\lambda)=0} \lambda P_\lambda, \qquad E = \sum_{f(\lambda)=0} P_\lambda.$$

ここで E は単位行列であり, 行列値関数 $R(x) = (A - xE)^{-1}$ を A のレゾルベントと呼ぶ.

さて, $R(x)$ については次が成り立つ.

$$(A - xE)^{-1} = -\frac{1}{f(x)} q(A, xE), \quad \text{ここで } q(x,y) = \frac{f(x) - f(y)}{x - y} \in \mathbf{Q}[x, y].$$

次に $f(x)$ が無平方であることから, $\gcd(f, f') = 1$ であるので, 拡張ユークリッド互除法により,

$$a(x)f(x) + b(x)f'(x) = 1$$

となる多項式 $a(x), b(x) \in \mathbf{Q}[x]$ が存在する. よって, 留数定理により

$$\begin{aligned} P_\lambda &= \frac{1}{2\pi\sqrt{-1}} \oint_\lambda (A - xE)^{-1} dx = -\frac{1}{2\pi\sqrt{-1}} \oint_\lambda \frac{1}{f(x)} q(A, xE) dx \\ &= -\frac{1}{2\pi\sqrt{-1}} \oint_\lambda \left\{ b(x) q(A, xE) \frac{f'(x)}{f(x)} + a(x) q(A, xE) \right\} dx \\ &= b(\lambda) q(A, \lambda E) \end{aligned}$$

以上より, 拡大体 $\mathbf{Q}(\lambda)$ の元を要素とする行列 P_λ は次の手順で求まることがわかる.

*ohara@air.s.kanazawa-u.ac.jp
†tajima@ie.niigata-u.ac.jp

アルゴリズム 1 入力: A, 出力: P_λ

1. 最小多項式 $f(x)$ の導出
2. 逆元計算 $b(x) \leftarrow f'(x)^{-1}$ (in $\mathbf{Q}[x]/\langle f(x)\rangle$)
3. 行列計算 $P_\lambda \leftarrow b(\lambda)q(A,\lambda E) \mod f(\lambda)$

さらに、最小多項式 $f(x)$ が $\mathbf{Q}[x]$ で可約な場合には, 次のようにして計算量を減らすことができる. $f(x) = f_1(x)\cdots f_m(x)$ とし,

$$q_i(x,y) = \frac{f_i(x) - f_i(y)}{x-y} \in \mathbf{Q}[x,y], \qquad h_i(x) = \frac{f(x)}{f_i(x)} \in \mathbf{Q}[x]$$

とすると, $f_i(x) = 0$ の根 λ_i に対し

$$\begin{aligned} P_{\lambda_i} &= -\frac{1}{2\pi\sqrt{-1}}\oint_{\lambda_i} \frac{1}{f_i(x)h_i(x)} q_i(A,xE)h_i(A)dx \\ &= -\frac{1}{2\pi\sqrt{-1}}\oint_{\lambda_i} b_i(x)q_i(A,xE)h_i(A)\frac{f_i'(x)}{f_i(x)}dx \\ &= b_i(\lambda_i)q_i(A,\lambda_i E)h_i(A) \end{aligned}$$

ここで $b_i(x) = (f_i'(x)h_i(x))^{-1}$ (in $\mathbf{Q}[x]/\langle f_i(x)\rangle$) である. したがってアルゴリズムは次のように書き表される.

アルゴリズム 2 入力: A, 出力: $\{P_{\lambda_1},\ldots,P_{\lambda_m}\}$

1. 最小多項式 $f(x)$ の導出
2. 因数分解 $f(x) = f_1(x)\cdots f_m(x)$
3. **for** $i = 1,\ldots,m$
   (a) 逆元計算 $b_i(x) = (f_i'(x)h_i(x))^{-1}$ (in $\mathbf{Q}[x]/\langle f_i(x)\rangle$)
   (b) 行列計算 $b_i(x)q_i(A,xE)h_i(A)$

因子の個数が並列度 N より多い場合, それぞれの因子毎に並列化可能. つまり, 逆元計算と行列計算を並列化できる.

2 並列化

まず, アルゴリズム 1, 2 の性能を見るために, Risa/Asir で実装し, アルゴリズムの各部分について, CPU 時間を測定した. ただし最小多項式の導出部分については, 木村欣司氏の固有多項式導出プログラム (C 言語による) を使用させていただいた.

実験には, Xeon (3.0GHz, 4 コア) × 2, 32GB メモリーの計算機と Linux kernel 2.6.18 を用い, 行列 A は, 64 次正方行列, 各要素は 18 ビット整数とした. CPU 時間は以下のようであった. また, 使用メモリはおよそ 6 GB であった.

この結果より, 行列多項式計算を高速化することが重要であることが分かったが, この部分は以下のように分散化することができるのである.

	(1) 最小多項式計算	(2) 逆元計算	(3) 行列多項式計算
Time	0.096s	38.3s	2522s

2.1 行列の多項式の分散計算

行列の多項式 $g(A) = g_0E + g_1A + \cdots + g_mA^m$ は各ブロック毎に分割して計算することができる. 行列 A を縦に ℓ 個のブロックに分割し, $A = \begin{pmatrix} A_1 \\ \vdots \\ A_\ell \end{pmatrix}$ とすると, 行列 $g(A)$ の i 番目のブロックは $(g(A))_i = g_0E_i + g_1A_i + g_2A_iA + \cdots + g_mA_iA^{m-1}$ で与えられる. ここで E_i は A の分割に対応した単位行列のブロックである. さらに多項式の計算に, ホーナー法を用いることで, $(g(A))_i$ は, ブロックと行列の $m-1$ 回の乗算で求められる.

この計算を効率的に CPU に割り当てるために, Risa/Asir 用分散計算パッケージ (oh_parallel.rr) を新規に開発した [2]. このパッケージの特徴は,

- Risa/Asir の通信機能 [1] を用いて, N 個の ox_asir を並列実行.
- M 個の仕事があるとき, ox_asir たちに仕事を勝手に割り当てていく. 手が空いたら自動的に次の仕事を割り当てる.

である. このパッケージを利用して, アルゴリズム 1 の 行列多項式計算 $b(x)q(A, xE) \mod f(x)$ を並列化した. なお, 並列化にあたって使用メモリを最小限に抑えるため, 各ブロックは行ベクトルとした.

2.2 実験結果 (固有多項式が既約)

アルゴリズム 1 に基づいて、逐次計算と並列計算のプログラムを作成し比較した. 以下に逐次プログラムと並列プログラムの経過時間のデータを示す. 実験環境は先に示したものと同じである. また, n は正方行列 A の次数である. A の各要素は 18 ビット整数とした.

n	24	32	40	48	56	64	72
逐次計算	3.38	24.60	98.93	326.44	891.04	2561.0	5522.4
並列計算	1.59	8.63	30.92	97.26	246.22	595.80	1299.67
比	2.12	2.85	3.20	3.36	3.62	4.30	4.25

実験環境はコア数 4 の計算機である. 一見して分かるように n が大きくなるにつれ, 並列化効率が向上していることが分かる. これは計算全体における行列多項式計算の比重が高まることを示している. さらに, 注目すべきは $n \geq 64$ ではコア数 4 を越えて, super linear になっていることである. これは通信時間のコストよりも, Risa/Asir のガーベージコレクタのコストの方が大きくなっていることを意味しているとも考えられる.

2.3 実験結果 (最小多項式が可約かつ無平方)

アルゴリズム 2 では、因子の個数が並列度 N より多い場合, それぞれの因子毎に並列化可能である. つまり, 逆元計算と行列計算を並列化できる. ここではアルゴリズム 2 に基づいて、逐次計算と並列計算のプ

ログラムを作成し比較した. 以下に逐次プログラムと並列プログラムの経過時間のデータを示す. 実験環境は先に示したものと同じである. また, 正方行列 A の次数は n, A の各要素は 8 ビット整数とし, 最小多項式の因子数を 8, 各因子の次数が $n/8$ であるような条件のもとで計測した.

n	96	104	112	120	128	136
逐次計算	39.49	60.74	90.54	117.8	176.3	234.3
並列計算	14.10	21.14	31.56	42.10	58.9	82.8

3 固有ベクトル計算

スペクトル分解 P_λ の各列ベクトルは, 固有ベクトルにもなっている. 固有多項式が既約な場合はどの列を計算してもよい. アルゴリズム 1 をもとに固有ベクトルを逐次計算し, 各部分の所要時間を調べた. 行列サイズ 64×64, 各要素 18 bits 整数のある行列の場合には次の表のようになった.

	(1) 最小多項式計算	(2) 逆元計算	(3) 行列多項式計算
Time	0.117s	39.53s	34.30s

この結果から逆元計算のウェイトが大きく, 並列化を試みてもそのままでは台数効果が期待しにくいことが予想できる. 我々は最小多項式が可約な場合にプログラムを作成して実験を行った.

正方行列 A の次数は n, A の各要素は 8 ビット整数とし, 最小多項式の因子数を 8, 各因子の次数が $n/8$ である.

n	96	104	112	120	128	136	144
逐次	6.38	7.96	11.58	14.60	17.79	25.71	31.55
並列	3.18	4.27	5.42	8.09	10.17	13.58	17.92

4 結論

最小多項式が既約な場合では, 行列の次数が大きいときには並列化効率は線形に近づいており, このアルゴリズムが並列計算に向いていることを示している. さらに Risa/Asir では超線形になるが, これは通信時間のコストよりも, Risa/Asir のガーベージコレクタのコストの方が大きくなっていることを意味していると考えられる.

参 考 文 献

[1] 小原功任, 高山信毅, 田村恭士, 野呂正行, 前川将秀: OpenXM 1.1.3 の概要, 数理研講究録 **1199** (2001), 179–191.

[2] `http://www.math.sci.kobe-u.ac.jp/cgi/cvsweb.cgi/OpenXM/src/asir-contrib/packages/src/oh_parallel.rr`